AF/21358
PATENT
IFW

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Roger Kenneth Abrams
Serial No.: 09/708,397 Art Unit: 2135
Filed: November 8, 2000 Examiner: Thanhnga Truong
For: SYSTEM AND METHOD FOR PREVENTION OF BUFFER OVERFLOW INTRUSIONS

Mail Stop Appeal Brief-Patents

Commissioner for Patents

P.O. Box 1450

Alexandria, VA 22313-1450

**TRANSMITTAL OF APPEAL BRIEF
(PATENT APPLICATION - 37 CFR 1.192)**

1. Transmitted herewith in triplicate is the APPEAL BRIEF in this application with respect to the Notice of Appeal filed on July 22, 2004.

NOTE: "The appellant shall, within 2 months from the date of the notice of appeal under § 1.191 in an application, reissue application, or patent under reexamination, or within the time allowed for response to the action appealed from, if such time is later, file a brief in triplicate." 37 CFR 1.192(a) (emphasis added).

2. STATUS OF APPLICANT

This application is on behalf of

☒ other than a small entity☐ small entity

verified statement:

☐ attached☐ already filed**3. FEE FOR FILING APPEAL BRIEF**

Pursuant to 37 CFR 1.17(f) the fee for filing the Appeal Brief is:

☐ small entity \$165.00☒ other than a small entity \$330.00**Appeal Brief fee due \$330.00****CERTIFICATE OF MAILING (37 CFR § 1.8)**

I hereby certify that this paper (along with any paper referred to as being attached or enclosed) is being deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Date: 8/20/04Serena Beller

(Type or print name of person mailing paper)

Serena Beller

(Signature of person mailing paper)

4. EXTENSION OF TERM

NOTE: The time periods set forth in 37 CFR 1.192(a) are subject to the provision of § 1.136 for patent applications. 37 CFR 1.191(d). Also see Notice of November 5, 1985 (1060 O.G. 27).

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136 apply.

(complete (a) or (b) as applicable)

- (a) ☐ Applicants petition for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d)) for the total number of months checked below:

Extension (months)	Fee for other than small entity	Fee for small entity
<input type="checkbox"/> one month	\$ 110.00	\$ 55.00
<input type="checkbox"/> two months	\$ 420.00	\$ 210.00
<input type="checkbox"/> three months	\$ 950.00	\$ 475.00
<input type="checkbox"/> four months	\$ 1,480.00	\$ 740.00
Fee		

If an additional extension of time is required, please consider this a petition therefor.

(check and complete the next item, if applicable)

- ☐ An extension for _____ months has already been secured and the fee paid therefor of \$ _____ is deducted from the total fee due for the total months of extension now requested.

Extension fee due with this request \$ _____
or

- (b) ☒ Applicants believe that no extension of term is required. However, this conditional petition is being made to provide for the possibility that applicants have inadvertently overlooked the need for a petition and fee for extension of time.

5. TOTAL FEE DUE

The total fee due is:

Appeal Brief fee \$330.00

Extension fee (if any) \$0

TOTAL FEE DUE \$330.00

6. FEE PAYMENT

- ☐ Attached is a check in the sum of \$ _____

- ☒ Charge Account No. 50-0563 (RPS9-2000-0077-US1) the sum of \$330.00.

A duplicate of this transmittal is attached.

7. FEE DEFICIENCY

NOTE: If there is a fee deficiency and there is no authorization to charge an account, additional fees are necessary to cover the additional time consumed in making up the original deficiency. If the maximum, six-month period has expired before the deficiency is noted and corrected, the application is held abandoned. In those instances where authorization to charge is included, processing delays are encountered in returning the papers to the PTO Finance Branch in order to apply these charges prior to action on the cases. Authorization to charge the deposit account for any fee deficiency should be checked. See the Notice of April 7, 1986, 1065 O.G. 31-33.

- ☒ If any additional extension and/or fee is required, this is a request therefor and to charge Account No. 50-0563 (RPS9-2000-0077-US1).

AND/OR

- ☒ If any additional fee for claims is required, charge Account No. 50-0563 (RPS9-2000-0077-US1).

Reg. No.: 47,159


SIGNATURE OF ATTORNEY OR PATENT AGENT

Tel. No.: (512) 370-2832

Robert A. Voigt, Jr.
WINSTEAD SECHREST & MINICK P.C.
P.O. Box 50784
Dallas, Texas 75201

Austin_1 258347v.1

RPS9-2000-0077US1

PATENT



- 1 -

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:	:	Before the Examiner:
Roger Kenneth Abrams	:	Truong, Thanhnga
Serial No.: 09/708,397	:	Group Art Unit: 2135
Filed: November 8, 2000	:	
	:	IBM Corporation
Title: SYSTEM AND METHOD FOR	:	P.O. Box 12915
PREVENTION OF BUFFER OVERFLOW:	:	Dept. 9CCA, Bldg. 002
INTRUSIONS	:	Research Triangle Park, NC 27709

APPEAL BRIEF

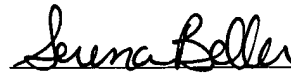
Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

I. **REAL PARTY IN INTEREST**

The real party in interest is International Business Machines Corporation, which is the assignee of the entire right, title and interest in the above-identified patent application.

CERTIFICATION UNDER 37 C.F.R. § 1.8

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450, on August 20, 2004.



Signature

08/25/2004 HGUTEMAI 00000003 500563 09708397

01 FC:1402 330.00 DA

Serena Beller
(Printed name of person certifying)

II. RELATED APPEALS AND INTERFERENCES

There are no other appeals or interferences known to Appellant, Appellant's legal representative or assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. STATUS OF CLAIMS

Claims 1-25 are pending in the Application. Claims 1-25 stand rejected.

IV. STATUS OF AMENDMENTS

The Appellant's response to the Office Action having a mailing date of January 15, 2004, has been considered, but the Examiner indicated that it did not place the application in condition for allowance because Appellant's arguments were deemed unpersuasive.

V. SUMMARY OF INVENTION

The increase in connectivity between computers has led to a proliferation of interface data processing and computing devices in a large network, commonly referred to as the "Internet." Specification, page 1, lines 9-11. The interconnection of the large number of resources has had many benefits, key among them the ability to share resources between a number of remotely-located processing machines. Specification, page 1, lines 11-13.

Key among the disadvantages of such wide-area networked schema, however, are the security issues raised by the increased access of computing devices to the entire membership of the Internet. Specification, page 1, lines 14-16. As the Internet grows, access for the membership of the Internet becomes increasingly synonymous with access for the public at large. Specification, page 1, lines 16-17.

In answer to this reality, many popular security measures have been instituted on data processing machines throughout the Internet. Specification, page 1, lines 18-19. These security measures range from the very simple (e.g., requiring a user name and password as a precondition for access to a machine) to the complex (e.g., construction of sophisticated firewalls which limit access to certain machines from certain addresses and through certain ports). Specification, page 1, line 19 – page 2, line 1. As the number of different security systems has increased, so have the opportunities for rogue users to attempt to access remote systems improperly. Specification, page 2, lines 1-3. These attempts, commonly called "hacking," give rise to the need to design more sophisticated security systems to meet and defeat these attacks. Specification, page 2, lines 3-5.

One of the most common methods for compromising the security of a data processing machine in a networked environment is called a buffer overflow attack. Specification, page 2, lines 6-7.

Buffer overflow attacks may be perpetrated against a remote machine which accepts user commands and data remotely, called the "server". Specification, page 2, lines 8-9. The server receives the commands and data from a user's machine, called a "client". Specification, page 2, lines 9-10. Upon receipt, the server may store the commands and/or data received from the client in a temporary memory location, called the buffer. Specification, page 2, lines 10-12. In most server operating and memory management systems, the buffers made available for storing commands and data received from a client are within a data structure called the "stack". Specification, page 2, lines 12-14.

The stack is a contiguous block of logical memory space which is made available to an application or process for the storage of data. Specification, page 2, lines 15-16. The data stored by an application or process within the stack may comprise local variable, definitions and values, the definitions and values of

parameters passed to the application or process, a return address or instructions, as well as temporary buffers. Specification, page 2, lines 16-19.

Those that attack users, called "hackers", may attempt to use the fact that all these data items are stored within a contiguous memory structure to their advantage in infiltrating the security measures of an application or process. Specification, page 2, lines 20-22. A hacker wishing to infiltrate such a system may send a block of data from a client to the server where the data is longer than the application or process is expecting. Specification, page 2, lines 22-24. The server for the application or process stores the data within the buffer on the stack. Specification, page 2, lines 24-25. However, the server places the data in a buffer sized to receive a normal data block. Specification, page 2, line 25 – page 3, line 1. The result is that the data is written past the end of the buffer. Specification, page 3, lines 1-2. On a server machine having a stack architecture, this overflow results in the portion of the stack holding the application's or process' other data, being overwritten. Specification, page 3, lines 2-4. Notably, the return address for the application or process may be some of the very data that is overwritten. Specification, page 3, lines 4-5. A clever hacker can design such a buffer overflow so that a process' return address is overwritten with a reference to programming code placed in the buffer or an overwritten portion of the stack. Specification, page 3, lines 5-7.

In such a case, when the process owning the stack ceases execution and attempts to return to a calling application or process, the return address it executes causes an infiltrator's code to execute instead. Specification, page 3, lines 8-10. The infiltrating code may create new user IDs with super-user authority, delete files, copy files, or provide other security breaches to the benefit of the hacker. Specification, page 3, lines 10-12.

Once the system is compromised in this manner, a hacker may fully compromise the system, removing suspicious entries from logging files and

modifying programs to introduce very sophisticated subversive access routes (called "back doors"), resulting in a fully compromised system. Specification, page 3, lines 13-16.

What is needed is a system and method for improved security on data processing archives to prevent these types of buffer overflow attacks. Specification, page 3, lines 17-18. Such a system and method should provide the flexibility of access to authorized users of the server machine while denying hackers the ability to place inappropriate return addresses and executable code on the process stack. Specification, page 3, lines 18-21. What is also needed is a system and method which performs these tasks so that undue overhead and expense in processor time and memory. Specification, page 3, lines 21-22. Several authors in the art have suggested placing a checking variable adjacent to the return address within the stack in order to thwart these attacks. Specification, page 3, lines 22-24. The checking variable, called a "canary," can be checked prior to any return address call in order to ensure that the stack has not been overwritten. Specification, page 3, line 24 – page 4, line 1. If the stack has been overwritten, the value of the canary will be changed and execution of the application may be aborted before an inappropriate return call is made. Specification, page 4, lines 1-3.

The deficiency in the canary approach is that it adds excessive instructions to verify the canary upon each return address call. Specification, page 4, lines 4-5. Accordingly, what is needed is a mechanism to prevent buffer overflow attacks which does not cause such an increase in processing time. Specification, page 4, lines 5-7.

The present invention addresses the foregoing needs by providing a system and method for protecting stacks against buffer overflow attacks. Specification, page 5, lines 2-3. The present invention takes advantage of Application Programming Interfaces ("APIs") of modern applications to create a new form of adaptation to

prevent the execution of rogue programming code introduced during a buffer overflow attack. Specification, page 5, lines 3-6.

In one embodiment of the present invention, a user-space accessible API is added to the memory management that can change the "execute-on-stack" capability of the memory model on a per-process basis. Specification, page 5, lines 7-9. Accordingly, a process that begins which is known to be subject to these kinds of buffer overflow attacks can call the API indicating to the underlying operating system and memory manager that the application is disallowing any execute-on-stack capability for itself. Specification, page 5, lines 9-12. The memory manager or operating system records this fact in the process structure. Specification, page 5, lines 12-13. The operating system and/or memory manager can mark the process stack structure accordingly. Specification, page 5, lines 13-14.

When the memory manager or the operating system receives a page fault and attempts to allocate a new page of memory, it looks to the process structure for the current process to determine if execute on stack capability is TRUE or FALSE. Specification, page 5, lines 15-17. If the process has not called the API to disable execute on stack, then the execute on stack capability will be TRUE, and execution and memory allocation proceeds in a normal fashion. Specification, page 5, lines 17-20. No additional overhead is used. Specification, page 5, line 20.

If, on the other hand, the process has called the API causing the execute-on-stack flag to be set to FALSE, the memory manager will execute a routine in accordance with the present invention to determine whether the page to be allocated is a stack page. Specification, page 5, lines 21-23. If the page is to be a stack, the memory manager resets the bit in the page table that permits execute privileges for that page. Specification, page 5, lines 23-25. If a hacker attacks with a buffer overflow attack and the process executes its return instruction, attempts to execute code from the corrupted stack area will immediately be prohibited by the hardware.

Specification, page 5, line 25 – page 6, line 2. The hardware will trigger an interrupt that causes the process to be killed without executing any of the hacker's code. Specification, page 6, lines 2-3.

VI. ISSUE

Are claims 1-25 properly rejected under 35 U.S.C. §102(e) as being anticipated by Moran (U.S. Patent No. 6,647,400)?

VII. GROUPING OF CLAIMS

Claims 1 and 18 form a first group.

Claims 2, 10 and 19 form a second group.

Claims 3-5, 11-12 and 20-21 form a third group.

Claims 6-8, 15-17 and 22-25 form a fourth group.

Claims 13 and 14 form a fifth group.

Claim 9 should not be grouped together and should be considered separately.

The reasons for these groupings are set forth in Appellant's arguments in Section VIII.

VIII. ARGUMENT

The Examiner has rejected claims 1-25 under 35 U.S.C. §102(e) as being anticipated by Moran. Paper No. 7, page 2. Appellant respectfully traverses these rejections for at least the reasons stated below.

For a claim to be anticipated under 35 U.S.C. §102, each and every claim limitation must be found within the cited prior art reference and arranged as required by the claim. M.P.E.P. §2131.

Appellant respectfully asserts that Moran does not disclose "a RAM connected to the bus system, the RAM being divided into pages, each page having an execution flag" as recited in claim 1 and similarly in claims 9 and 18. The Examiner cites column 5, lines 53-55; column 6, lines 9-12 and column 6, line 52 – column 7, line 1 of Moran as disclosing the above-cited claim limitation. Paper No. 7, page 8. Appellant respectfully traverses and asserts that Moran instead discloses the different types of computer readable mediums as well as the fact that information retained within mass storage units 112, 120 may be incorporated, if needed, in standard fashion as part of the primary storage unit, e.g., storage unit 110 such as RAM, as virtual memory. Moran further discloses that the primary storage unit typically includes basic operating instructions, program code, data and objects used by the central processing unit to perform its functions. The Examiner then asserts that program code necessarily discloses an execution flag. Paper No. 7, page 8. Appellant respectfully traverses the assertion that program code necessarily discloses an execution flag. The Examiner must provide a basis in fact and/or technical reasoning to reasonably support the determination that "program code" as used in Moran discloses an execution flag. *Ex parte Levy*, 17 U.S.P.Q.2d 1461, 1464 (Bd. Pat. App. & Inter. 1990). That is, the Examiner must provide extrinsic evidence that must make clear that "program code" as used in Moran discloses an execution flag. *In re Robertson*, 169 F.3d 743, 745, 49 U.S.P.Q.2d 1949, 1950-51 (Fed. Cir. 1999). Therefore, the Examiner must provide objective evidence and not rely on her own subjective opinion to support the assertion that "program code" as used in Moran discloses an execution flag. *See In re Lee*, 61 U.S.P.Q.2d 1430, 1434 (Fed. Cir. 2002). However, the Examiner has not provided any objective evidence to support her assertion that "program code" as used in Moran discloses an execution flag. Thus, Moran does not disclose all of the limitations of claims 1, 9 and 18, and thus Moran does not anticipate claims 1, 9 and 18. M.P.E.P. §2131.

Appellant further asserts that Moran does not disclose "a memory manager configured to manage the pages of the RAM and permit CPU execution of data on pages according to the execution flag" as recited in claim 1 and similarly in claims 9 and 18. The Examiner cites column 5, lines 25-58 and column 38, lines 35-40 of Moran as disclosing the above-cited claim limitation. Paper No. 7, pages 9-10. Appellant respectfully traverses and asserts that Moran instead discloses a general purpose computer system. Column 5, lines 28-58. Moran further discloses an architecture of an intrusion detection system which includes an analysis engine (element 302 of Figure 3) and an event database (element 304 of Figure 3). Column 8, lines 9-10. Moran further discloses that the analysis engine utilizes rules and an attack signatures database and receives input from a sensor controller. Column 8, lines 10-12. Moran further discloses, with respect to the analysis engine, that on a multitasking operating system, processes and threads both allow interleaving of flow of control, both allowing the user of the processor to switch from a computation that has reached a point where it can no longer proceed to one that is ready to run. Column 38, lines 35-40. The Examiner then asserts that an operating system necessarily discloses the above-cited claim limitation. Paper No. 7, page 10. Appellant respectfully traverses the assertion that an operating system necessarily manages the pages of RAM and permits CPU execution of data on pages according to an execution flag. There is no language in the cited passages of Moran disclosing an execution flag. Neither is there any language in the cited passages of Moran that discloses permitting CPU execution of data on pages according to an execution flag. The Examiner must provide a basis in fact and/or technical reasoning to reasonably support the determination that an "operating system" as used in Moran discloses the above-cited claim limitation. *Ex parte Levy*, 17 U.S.P.Q.2d 1461, 1464 (Bd. Pat. App. & Inter. 1990). That is, the Examiner must provide extrinsic evidence that must make clear that an "operating system" as used in Moran discloses the above-cited claim limitation. *In re Robertson*, 169 F.3d 743, 745, 49 U.S.P.Q.2d 1949, 1950-51 (Fed.

Cir. 1999). Therefore, the Examiner must provide objective evidence and not rely on her own subjective opinion to support the assertion that an "operating system" as used in Moran discloses the above-cited claim limitation. *See In re Lee*, 61 U.S.P.Q.2d 1430, 1434 (Fed. Cir. 2002). However, the Examiner has not provided any objective evidence to support her assertion that an "operating system" as used in Moran discloses the above-cited claim limitation. Thus, Moran does not disclose all of the limitations of claims 1, 9 and 18, and thus Moran does not anticipate claims 1, 9 and 18. M.P.E.P. §2131.

Appellant further asserts that Moran does not disclose "wherein the memory manager is configured to determine whether the program is susceptible to buffer overflow attacks, and, if so, set the execution flag for program stack pages of RAM to deny CPU execution of data on the program stack pages of RAM" as recited in claim 1 and similarly in claims 9 and 18. The Examiner cites column 33, lines 64-67; column 34, lines 1-3; and column 34, lines 43-50 of Moran as disclosing the above-cited claim limitation. Paper No. 7, pages 3-4. Appellant respectfully traverses and asserts that Moran instead discloses that a SetUID comes from the operating system setting the UID of the command's process to be that of the owner of the file. Column 34, lines 5-7. Moran further discloses that SetUID commands provide restricted access to system resources. Column 34, lines 8-10. Moran further discloses that almost all buffer overflow attacks take effect at the very beginning of the execution of the program, because the data causing the overflow is supplied as part of the command invocation or setup. Column 34, lines 43-46. Moran further discloses that the command is subverted (replaced) before it has a chance to perform any of its intended actions. Column 34, lines 46-48. Moran further discloses that the analysis engine examines the last access time of each SetUID command. Column 34, lines 54-55. Moran further discloses that this access time is compared to the timestamps on files that the command is expected to access. Column 34, lines 60-61. Moran further discloses that if those timestamps are earlier than the last-access time

on the SetUID command, this is evidence that a SetUID buffer overflow attack may have occurred. Column 34, lines 61-64.

Hence, Moran discloses a description of an overflow buffer attack and a method of detecting whether an overflow attack may have occurred by comparing the last-access time of each SetUID command with the timestamps on files that the command is expected to access. There is no language in the cited passages of Moran disclosing a memory manager that is configured to determine whether a program is susceptible to buffer overflow attacks. Instead, Moran discloses a method of detecting whether a buffer overflow attack may have occurred. Further, there is no language in the cited passages of Moran that discloses setting an execution flag to deny CPU execution of data on the program stack pages of RAM. Thus, Moran does not disclose all of the limitations of claims 1, 9 and 18, and thus Moran does not anticipate claims 1, 9 and 18. M.P.E.P. §2131.

Appellant further asserts that Moran does not disclose "wherein the memory manager and the CPU are configured to deny CPU execution of data by triggering a hardware interrupt" as recited in claim 2 and similarly in claims 10 and 19. As understood by the Appellant, the Examiner asserts that it is inherent that memory 110 of Moran includes a memory manager which along with CPU 102 are configured to deny CPU execution of data by triggering a hardware interrupt. The Examiner does cite to column 5, lines 43-55 of Moran. Paper No. 7, page 4. However, upon review of the cited passage, there is no language in the cited passage that discloses hardware interrupts, denying CPU execution or a memory manager. Appellant respectfully traverses the assertion that Moran inherently discloses a memory manager and a CPU configured to deny CPU execution of data by triggering a hardware interrupt. The Examiner must provide a basis in fact and/or technical reasoning to reasonably support the determination that memory 110 of Moran inherently discloses a memory manager and a CPU configured to deny CPU execution of data by triggering a hardware interrupt. *Ex parte Levy*, 17 U.S.P.Q.2d 1461, 1464 (Bd. Pat. App. & Inter.

1990). That is, the Examiner must provide extrinsic evidence that must make clear that memory 110 of Moran inherently discloses a memory manager and a CPU configured to deny CPU execution of data by triggering a hardware interrupt. *In re Robertson*, 169 F.3d 743, 745, 49 U.S.P.Q.2d 1949, 1950-51 (Fed. Cir. 1999). Therefore, the Examiner must provide objective evidence and not rely on her own subjective opinion to support the assertion that memory 110 of Moran inherently discloses a memory manager and a CPU configured to deny CPU execution of data by triggering a hardware interrupt. *See In re Lee*, 61 U.S.P.Q.2d 1430, 1434 (Fed. Cir. 2002). As the Examiner has not provided any objective evidence for supporting that memory 110 inherently includes a memory manager and a CPU configured to deny CPU execution of data by triggering a hardware interrupt, the Examiner has not presented a *prima facie* case of anticipation for rejecting claims 2, 10 and 19. M.P.E.P. §2131.

Appellant further asserts that Moran does not disclose "a process structure table in data communication with the memory manager, wherein the memory manager comprises an annotation API, wherein the annotation API is configured to annotate within the process structure table the susceptibility of the program to buffer overflow attacks, and wherein the memory manager is configured to make the determination of susceptibility to buffer overflow attacks with reference to the process structure table" as recited in claim 3 and similarly in claims 4-5, 11-12 and 20-21. As understood by the Appellant, the Examiner asserts that the above-cited claim limitations are inherent in Moran using language cited in column 5, lines 43-55 of Moran. Paper No. 7, pages 4-5. Appellant respectfully traverses the assertion that the above-cited claim limitations are inherently disclosed in Moran. Further, upon review of the cited passage, there is no language in the cited passage that discloses a process structure table, a memory manager, an API, and an API configured to annotate within the process structure table the susceptibility of the program to buffer overflow attacks or a memory manager configured to make the determination of

susceptibility to buffer overflow attacks with reference to the process structure table. As stated above, the Examiner must provide a basis in fact and/or technical reasoning to reasonably support the determination that Moran inherently discloses the above-cited claim limitations. *Ex parte Levy*, 17 U.S.P.Q.2d 1461, 1464 (Bd. Pat. App. & Inter. 1990). That is, the Examiner must provide extrinsic evidence that must make clear that Moran inherently discloses the above-cited claim limitations. *In re Robertson*, 169 F.3d 743, 745, 49 U.S.P.Q.2d 1949, 1950-51 (Fed. Cir. 1999). As the Examiner has not provided any objective evidence supporting her inherency arguments, the Examiner has not presented a *prima facie* case of anticipation for rejecting claims 3-5, 11-12 and 20-21. M.P.E.P. §2131.

Appellant further asserts that Moran does not disclose "wherein the program is configured to call the annotation API if the program is susceptible to buffer overflow attacks, the memory manager is configured to determine susceptibility upon a request to allocate an additional page of RAM for the program" as recited in claim 6 and similarly in claims 7-8, 15-17 and 22-25. As understood by the Appellant, the Examiner asserts that memory 110 of Moran inherently discloses a memory manager configured to determine susceptibility upon a request to allocate an additional page of RAM for the program. Appellant respectfully traverses the assertion that Moran inherently discloses the above-cited claim limitation. The Examiner must provide a basis in fact and/or technical reasoning to reasonably support the determination that memory 110 of Moran inherently discloses a memory manager configured to determine susceptibility upon a request to allocate an additional page of RAM for the program. *Ex parte Levy*, 17 U.S.P.Q.2d 1461, 1464 (Bd. Pat. App. & Inter. 1990). That is, the Examiner must provide extrinsic evidence that must make clear that memory 110 of Moran inherently discloses a memory manager configured to determine susceptibility upon a request to allocate an additional page of RAM for the program. *In re Robertson*, 169 F.3d 743, 745, 49 U.S.P.Q.2d 1949, 1950-51 (Fed. Cir. 1999). As the Examiner has not provided any objective evidence in supporting the

assertion that memory 110 of Moran inherently discloses a memory manager configured to determine susceptibility upon a request to allocate an additional page of RAM for the program, the Examiner has not presented a *prima facie* case of anticipation for rejecting claims 6-8, 15-17 and 22-25. M.P.E.P. §2131.

Appellant further asserts that Moran does not disclose "an application program code comprising a set of codes operable to direct a data processing system to request the memory manager code to establish a program stack within at least one page the RAM" as recited in claim 9. As understood by the Appellant, the Examiner asserts that it is inherent in memory 110 of Moran to perform the above-cited claim limitation. Appellant respectfully traverses the assertion that memory 110 of Moran inherently includes codes to perform the above-cited claim limitation. There is no language in Moran that discloses codes operable to direct a data processing system to request a memory manager code to establish a program stack within at least one page within the RAM. The Examiner must provide a basis in fact and/or technical reasoning to reasonably support the determination that memory 110 of Moran inherently includes code to perform the above-cited claim limitation. *Ex parte Levy*, 17 U.S.P.Q.2d 1461, 1464 (Bd. Pat. App. & Inter. 1990). That is, the Examiner must provide extrinsic evidence that must make clear that memory 110 of Moran inherently includes codes to perform the above-cited claim limitation. *In re Robertson*, 169 F.3d 743, 745, 49 U.S.P.Q.2d 1949, 1950-51 (Fed. Cir. 1999). Therefore, the Examiner must provide objective evidence and not rely on her own subjective opinion to support the assertion that memory 110 of Moran inherently discloses the above-cited claim limitation. *See In re Lee*, 61 U.S.P.Q.2d 1430, 1434 (Fed. Cir. 2002). As the Examiner has not provided any objective evidence in supporting the assertion that memory 110 of Moran inherently includes code to perform the above-cited claim limitation, the Examiner has not presented a *prima facie* case of anticipation for rejecting claim 9. M.P.E.P. §2131.

Appellant further asserts that Moran does not disclose "wherein the memory manager code comprises the process structure table code as an API" as recited in claims 13-14. As understood by the Appellant, the Examiner asserts that memory 110 of Moran inherently discloses the above-cited claim limitation. Appellant respectfully traverses the assertion that memory 110 of Moran inherently discloses memory manager code that includes a process structure table code as an API. The Examiner must provide a basis in fact and/or technical reasoning to reasonably support the determination that memory 110 inherently discloses memory manager code that comprises a process structure table code as an API. *Ex parte Levy*, 17 U.S.P.Q.2d 1461, 1464 (Bd. Pat. App. & Inter. 1990). That is, the Examiner must provide extrinsic evidence that must make clear that memory 110 inherently discloses memory manager code that comprises a process structure table code as an API. *In re Robertson*, 169 F.3d 743, 745, 49 U.S.P.Q.2d 1949, 1950-51 (Fed. Cir. 1999). As the Examiner has not provided any objective evidence to support the assertion that memory 110 of Moran inherently discloses a memory manager code that comprises a process structure table code as an API, the Examiner has not presented a *prima facie* case of anticipation for rejecting claims 13-14. M.P.E.P. §2131.

As a result of the foregoing, Appellant respectfully asserts that not each and every claim limitation was found within the cited prior art reference, and thus claims 1-25 are not anticipated by Moran.

IX. CONCLUSION

For the reasons noted above, the rejections of claims 1-25 are in error. Appellant respectfully requests reversal of the rejections and allowance of claims 1-25.

Respectfully submitted,

WINSTEAD SECHREST & MINICK P.C.

Attorneys for Appellant

By: _____

Robert A. Voigt, Jr.
Reg. No. 47,159
Kelly K. Kordzik
Reg. No. 36,571

P.O. Box 50784
Dallas, Texas 75201
(512) 370-2832

APPENDIX

1. A data processing system comprising:
 - a bus system;
 - a CPU connected to the bus system;
 - a RAM connected to the bus system, the RAM being divided into pages, each page having an execution flag;
 - a memory manager configured to manage the pages of the RAM and permit CPU execution of data on pages according to the execution flag;
 - a program stored within at least one page of the RAM; and
 - a program stack stored within at least one page the RAM,wherein the memory manager is configured to determine whether the program is susceptible to buffer overflow attacks, and, if so, set the execution flag for program stack pages of RAM to deny CPU execution of data on the program stack pages of RAM.
2. The data processing system of claim 1 wherein the memory manager and the CPU are configured to deny CPU execution of data by triggering a hardware interrupt.
3. The data processing system of claim 1 further comprising:
 - a process structure table in data communication with the memory manager,
 - wherein the memory manager comprises an annotation API,
 - wherein the annotation API is configured to annotate within the process structure table the susceptibility of the program to buffer overflow attacks, and
 - wherein the memory manager is configured to make the determination of susceptibility to buffer overflow attacks with reference to the process structure table.
4. The data processing system of claim 2 further comprising:

a process structure table in data communication with the memory manager,
wherein the memory manager comprises an annotation API,
wherein the annotation API is configured to annotate within the process structure table the susceptibility of the program to buffer overflow attacks, and
wherein the memory manager is configured to make the determination of susceptibility to buffer overflow attacks with reference to the process structure table.

5. The data processing system of claim 1 further comprising:
a process structure table in data communication with the memory manager,
and
an annotation program in data communication with the process structure table,
wherein the annotation program is configured to annotate within the process structure table the susceptibility of the program to buffer overflow attacks, and
wherein the memory manager is configured to make the determination of susceptibility to buffer overflow attacks with reference to the process structure table.
6. The data processing system of claim 3 wherein the program is configured to call the annotation API if the program is susceptible to buffer overflow attacks,
the memory manager is configured to determine susceptibility upon a request to allocate an additional page of RAM for the program.
7. The data processing system of claim 4 wherein the program is configured to call the annotation API if the program is susceptible to buffer overflow attacks,
the memory manager is configured to determine susceptibility upon a request to allocate an additional page of RAM for the program.
8. The data processing system of claim 5 wherein the program is configured to call the annotation program if the program is susceptible to buffer overflow attacks,
the memory manager is configured to determine susceptibility upon a request to allocate an additional page of RAM for the program.

9. A computer program product in a computer-readable medium adapted to prevent buffer overflow attacks comprising:

a memory manager code comprising a set of codes operable to direct a data processing system to manage a set of pages within a RAM of the data processing system and to permit a CPU of the data processing system to execute data on pages according to an execution flag on each of the set of pages;

an application program code comprising a set of codes operable to direct a data processing system to request the memory manager code to establish a program stack within at least one page the RAM; and

a susceptibility code comprising a set of codes operable to direct a data processing system to determine whether the application program code is susceptible to buffer overflow attacks, and, if so, set the execution flag for the program stack pages to deny CPU execution of data on the program stack pages.

10. The computer program product of claim 9 wherein the memory manager code further comprises a set of codes operable to direct a data processing system to deny CPU execution of data by triggering a hardware interrupt.

11. The computer program product of claim 9 further comprising:

a process structure table code comprising a set of codes operable to direct a data processing system to establish and maintain a process structure table code in data communication with the memory manager code and to annotate within the process structure table the susceptibility of the application program code to buffer overflow attacks,

wherein the memory manager code further comprises codes operable to direct a data processing system to make the determination of susceptibility to buffer overflow attacks with reference to the process structure table.

12. The computer program product of claim 10 further comprising:

a process structure table code comprising a set of codes operable to direct a data processing system to establish and maintain a process structure table code in data communication with the memory manager code and to annotate within the process structure table the susceptibility of the application program code to buffer overflow attacks,

wherein the memory manager code further comprises codes operable to direct a data processing system to make the determination of susceptibility to buffer overflow attacks with reference to the process structure table.

13. The computer program product of claim 11 wherein the memory manager code comprises the process structure table code as an API.

14. The computer program product of claim 12 wherein the memory manager code comprises the process structure table code as an API.

15. The computer program product of claim 11 wherein the application program code further comprises a set of codes operable to direct a data processing system to call the process structure table code the application program code is susceptible to buffer overflow attacks, and

the memory manager code further comprises a set of codes operable to direct a data processing system to determine susceptibility upon receipt of a request to allocate an additional page of RAM for the application program code.

16. The computer program product of claim 14 wherein the application program code further comprises a set of codes operable to direct a data processing system to call the process structure table code the application program code is susceptible to buffer overflow attacks, and

the memory manager code further comprises a set of codes operable to direct a data processing system to determine susceptibility upon receipt of a request to allocate an additional page of RAM for the application program code.

17. The computer program product of claim 13 wherein the application program code further comprises a set of codes operable to direct a data processing system to call the process structure table code the application program code is susceptible to buffer overflow attacks, and

the memory manager code further comprises a set of codes operable to direct a data processing system to determine susceptibility upon receipt of a request to allocate an additional page of RAM for the application program code.

18. A method for handling buffer overflow attacks against an application program running on a data processing system, having a CPU and a RAM divided into pages, the method comprising the steps of:

designating an execution flag for each page of RAM allocated to a stack of the application program;

permitting CPU execution of data on pages of RAM according to the execution flag;

determining whether the application program is susceptible to buffer overflow attacks; and

if the application program is susceptible to buffer overflow attacks, setting the execution flag as to the pages of RAM allocated to the stack of the application program.

19. The method of claim 18 wherein step of permitting CPU execution comprises the steps of:

examining the execution flag on the page of RAM;

if the execution flag is set, triggering a hardware interrupt; and

otherwise executing the data on the page.

20. The method of claim 18 further comprising the steps of:

establishing a process structure table;

maintaining the process structure table by annotating within the process structure table the susceptibility of the application program to buffer overflow attacks, wherein the step of determining susceptibility to buffer overflow attacks is made with reference to the process structure table.

21. The method of claim 19 further comprising the steps of:
establishing a process structure table;
maintaining the process structure table by annotating within the process structure table the susceptibility of the application program to buffer overflow attacks, wherein the step of determining susceptibility to buffer overflow attacks is made with reference to the process structure table.
22. The method according to claim 20 wherein the step of maintaining the process structure table is done once at the beginning of execution of the application program, and
the step of determining susceptibility is performed upon each receipt of a request to allocate an additional page of RAM for the application program code.
23. The method according to claim 21 wherein the step of maintaining the process structure table is done once at the beginning of execution of the application program, and
the step of determining susceptibility is performed upon each receipt of a request to allocate an additional page of RAM for the application program code.
24. The method according to claim 22 wherein request to allocate an additional page of RAM is a request to allocate the additional page of RAM for the stack.
25. The method according to claim 23 wherein request to allocate an additional page of RAM is a request to allocate the additional page of RAM for the stack.